Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application and reflects the examiner's amendments of 12/26/2006:

Listing of Claims:

1. (previously presented):

   A method of translating a block of code from a source architecture that supports multiple-format registers to a target architecture that does not, the method comprising:

      determining a register format of a source register operated on by a source instruction in a source block of code, the register format including an input instruction format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and being used as an input of the source instruction, the input instruction format containing format of the source register expected by the source instruction, the output block format containing format of the source register after the source block of code is executed;

      detecting an instruction format inconsistency between the source register and a target register of a target architecture during a translation phase of a binary translation that translates the source block of code into a target block of code running in the target architecture;

      emitting a target instruction sequence corresponding to the source instruction into the target block of code;

      emitting a block inconsistency check code into a prefix of the target block of code; and

      emitting a format update code to update a format register associated register format into a suffix of the target block of code.

2. (original):

   The method of claim 1 wherein detecting the instruction format inconsistency comprises:

      comparing the output block format to the input instruction format if the output block format asserts an access status of the source register.

3. (original):

   The method of claim 2 further comprising:

      emitting a conversion code to convert the source register from the output block format to the input instruction format into the target block of code during the translation phase if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

4. (original):

The method of claim 1 further comprising:

updating an input block format and the output block format, the input block format containing format of the source register expected by the source block of code before execution.

5. (original):

The method of claim 4 wherein updating comprises:

setting the input block format and the output block format to the input instruction format if the output block format does not assert an access status of the source register.

6. (original):

The method of claim 4 wherein updating comprises:

setting the output block format to the input instruction format if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

7. (original):

The method of claim 4 wherein updating comprises:

setting the output block format to the output instruction format for the source register being used as output of the source instruction.

8. (cancelled)

9. (original):

The method of claim 1 wherein determining the register format comprises:

determining one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.

10. (original):

The method of claim 1 wherein emitting the block inconsistency check code comprises:

emitting the block inconsistency check code to be executed during an execution phase following the translation phase.

11. (currently amended):

An article of manufacture to translate a block of code from a source architecture that supports multiple-format registers to a target architecture that does not, the article of manufacture comprising a processor readable storage device including thereon sequences of instructions that, when executed, cause a computer processor to:

determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input instruction format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and being used as an input of the source instruction, the input instruction format containing format of the source register expected by the source instruction, the output block format containing format of the source register after the source block of code is executed;

detect an instruction format inconsistency between the source register and a target register of a target architecture during a translation phase of a binary translation that translates the source block of code into a target block of code running in the target architecture;

emit a target instruction sequence corresponding to the source instruction into the target block of code;

emit a block inconsistency check code into prefix of the target block of code; and

emit a format update code to update a format register associated register format into the suffix of the target block of code.

12. (previously presented):

The article of manufacture of claim 11 wherein causing the computer processor to detect the instruction format inconsistency comprises sequences of instructions that, when executed, cause the computer processor to:

compare the output block format to the input instruction format if the output block format asserts an access status of the source register.

13. (previously presented):

The article of manufacture of claim 12 further comprising sequences of instructions that, when executed, cause the computer processor to:

emit a conversion code to convert the source register from the output block format to the input instruction format into the target block of code during the translation phase if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

14. (previously presented):

The article of manufacture of claim 11 further comprising sequences of instructions that, when executed, cause the computer processor to:

update an input block format and the output block format, the input block format containing format of the source register expected by the source block of code before execution.

15. (previously presented):

    The article of manufacture of claim 11 wherein causing the computer processor to update comprises sequences of instructions that, when executed, cause the computer processor to:

    set the input block format and the output block format to the input instruction format if the output block format does not assert an access status of the source register.

16. (previously presented):

    The article of manufacture of claim 11 wherein causing the computer processor to update comprises sequences of instructions that, when executed, cause the computer processor to:

    set the output block format to the input instruction format if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

17. (previously presented):

    The article of manufacture of claim 11 wherein causing the computer processor to update comprises sequences of instructions that, when executed, cause the computer processor to:

    set the output block format to the output instruction format for the source register being used as output of the source instruction.

18. (cancelled)

19. (previously presented):

    The article of manufacture of claim 11 wherein causing the computer processor to determine the register format comprises sequences of instructions that, when executed, cause the computer processor to:

    determine one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.

20. (original):

    The article of manufacture of claim 11 wherein causing the computer processor to emit the block inconsistency check code comprises sequences of instructions that, when executed, cause the computer processor to:

    emit the block inconsistency check code to be executed during an execution phase following the translation phase.

21. (previously presented):

    A computer system to translate a block of code from a source architecture that supports multiple-format registers to a target architecture that does not, the computer system comprising:

    a processor; and

a memory coupled to the processor to store program code, the program code, when executed, causing the processor to:

determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input instruction format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and being used as an input of the source instruction, the input instruction format containing format of the source register expected by the source instruction, the output block format containing format of the source register after the source block of code is executed;

detect an instruction format inconsistency between the source register and a target register of a target architecture during a translation phase of a binary translation that translates the source block of code into a target block of code running in the target architecture;

emit a target instruction sequence corresponding to the source instruction into the target block of code;

emit a block inconsistency check code into prefix of the target block of code; and

emit a format update code to update a format register associated register format into the suffix of the target block of code.

22. (previously presented):

The computer system of claim 21 wherein the program code causing the processor to detect the instruction format inconsistency causes the processor to:

compare the output block format to the input instruction format if the output block format asserts an access status of the source register.

23. (previously presented):

The computer system of claim 22 wherein the program code further causing the processor to:

emit a conversion code to convert the source register from the output block format to the input instruction format into the target block of code during the translation phase if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

24. (previously presented):

The computer system of claim 21 wherein the program code further causing the processor to:

update an input block format and the output block format, the input block format containing format of the source register expected by the source block of code before execution.

25. (previously presented):

    The computer system of claim 21 wherein the program code causing the processor to update causes the processor to:

    set the input block format and the output block format to the input instruction format if the output block format does not assert an access status of the source register.

26. (previously presented):

    The computer system of claim 21 wherein the program code causing the processor to update causes the processor to:

    set the output block format to the input instruction format if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

27. (previously presented):

    The computer system of claim 21 wherein the program code causing the processor to update causes the processor to:

    set the output block format to the output instruction format for the source register being used as output of the source instruction.

28. (cancelled)

29. (previously presented):

    The computer system of claim 21 wherein the program code causing the processor to determine the register format causes the processor to:

    determine one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.

30. (previously presented):

    The computer system of claim 21 wherein the program code causing the processor to emit the block inconsistency check code causes the processor to:

    emit the block inconsistency check code to be executed during an execution phase following the translation phase.

31. (previously presented):

    A method of translating a block of code from a source architecture that supports multiple-format registers to a target architecture that does not, the method comprising:

    determining a register format of a source register operated on by a source instruction in a source block of code, the register format including an input block format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and a format register associated with the register format, the input block format containing format of the source register expected by the source block of code, the output block format containing format of the source register after the source block of code is executed;

detecting a block format inconsistency between the source register and a target register of a target architecture during an execution phase of a binary translation that translates the source block of code into a target block of code running in the target architecture;

emitting a target instruction sequence corresponding to the source instruction into the target block of code;

emitting a block inconsistency check code into a prefix of the target block of code; and

emitting a format update code to update a format register associated register format into a suffix of the target block of code.

32. (original):

The method of claim 31 wherein detecting the block format inconsistency comprises:

masking the format register with an input block format mask; and

comparing the masked format register with the input block format.

33. (original):

The method of claim 31 further comprising:

updating the format register upon exit of the target block of code.

34. (original):

The method of claim 33 wherein updating the format register comprises:

generating a first comparison result between the format register and the output block format;

masking the first comparison result by an output block format mask; and

generating a second comparison result between the format register and the masked first comparison result, the second comparison result corresponding to the updated format register.

35. (original):

The method of claim 32 further comprising:

executing a self-correcting code if the masked input block format is different than the input block format.

36. (original):

The method of claim 35 wherein executing comprises:

asserting a correction condition based on the format register and the input block format.

37. (original):

The method of claim 36 wherein asserting comprises:

comparing the format register to the input block format if the input block format asserts an access status of the source register.

38. (original):

The method of claim 36 further comprising:

converting the source register from format contained in the format register to format contained in the input block format; and

setting the format register to the input block format.

39. (original):

The method of claim 31 wherein determining the register format comprises:

determining one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.

40. (original):

The method of claim 38 wherein detecting the block format inconsistency comprises:

detecting the block format inconsistency during the execution phase that follows a translation phase in the binary translation.

41. (currently amended):

An article of manufacture to translate a block of code from a source architecture that supports multiple-format registers to a target architecture that does not, the article of manufacture comprising a processor readable storage device including thereon sequences of instructions that, when executed, cause a computer processor to:

determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input block format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and a format register associated with the register format, the input block format containing format of the source register expected by the source block of code, the output block format containing format of the source register after the source block of code is executed;

detect a block format inconsistency between the source register and a target register of a target architecture during an execution phase of a binary translation that translates the source block of code into a target block of code running in the target architecture;

emit a target instruction sequence corresponding to the source instruction into the target block of code;

emit a block inconsistency check code into prefix of the target block of code; and

emit a format update code to update a format register associated register format into the suffix of the target block of code.

42. (previously presented):

The article of manufacture of claim 41 wherein causing the computer processor to detect the block format inconsistency comprises sequences of instructions that, when executed, cause the computer processor to:

mask the format register with an input block format mask; and

compare the masked format register with the input block format.

43. (previously presented):

The article of manufacture of claim 41 further comprising sequences of instructions that, when executed, cause the computer processor to:

update the format register upon exit of the target block of code.

44. (previously presented):

The article of manufacture of claim 43 wherein causing the computer processor to update the format register comprises sequences of instructions that, when executed, cause the computer processor to:

generate a first comparison result between the format register and the output block format;

mask the first comparison result by an output block format mask; and

generate a second comparison result between the format register and the masked first comparison result, the second comparison result corresponding to the updated format register.

45. (previously presented):

The article of manufacture of claim 42 further comprising sequences of instructions that, when executed, cause the computer processor to:

execute a self-correcting code if the masked input block format is different than the input block format.

46. (previously presented):

The article of manufacture of claim 45 wherein causing the computer processor to execute comprises sequences of instructions that, when executed, cause the computer processor to:

assert a correction condition based on the format register and the input block format.

47. (previously presented):

The article of manufacture of claim 46 wherein causing the computer processor to assert comprises sequences of instructions that, when executed, cause the computer processor to:

compare the format register to the input block format if the input block format asserts an access status of the source register.

48. (previously presented):

The article of manufacture of claim 46 further comprising sequences of instructions that, when executed, cause the computer processor to:

convert the source register from format contained in the format register to format contained in the input block format; and

set the format register to the input block format.

49. (previously presented):

The article of manufacture of claim 41 wherein causing the computer processor to determine the register format comprises sequences of instructions that, when executed, cause the computer processor to:

determine one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.

50. (previously presented):

The article of manufacture of claim 48 wherein causing the computer processor to detect the block format inconsistency comprises sequences of instructions that, when executed, cause the computer processor to:

detect the block format inconsistency during the execution phase that follows a translation phase in the binary translation.

51. (previously presented):

A computer system to translate a block of code from a source architecture that supports multiple-format registers to a target architecture that does not, the computer system comprising:

a processor; and

a memory coupled to the processor to store program code, the program code, when executed, causing the processor to:

determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input block format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and a format register associated with the register format, the input block format containing format of the source register expected by the source block of code, the output block format containing format of the source register after the source block of code is executed;

detect a block format inconsistency between the source register and a target register of a target architecture during an execution phase of a binary translation that translates the source block of code into a target block of code running in the target architecture;

emit a target instruction sequence corresponding to the source instruction into the target block of code;

emit a block inconsistency check code into prefix of the target block of code; and

emit a format update code to update a format register associated register format into the suffix of the target block of code.

52. (previously presented):

The computer system of claim 51 wherein the program code causing the processor to detect the block format inconsistency causes the processor to:

mask the format register with an input block format mask; and

compare the masked format register with the input block format.

53. (previously presented):

The computer system of claim 51 the program code further causing the processor to:

update the format register upon exit of the target block of code.

54. (previously presented):

The computer system of claim 53 wherein the program code causing the processor to update the format register causes the processor to:

generate a first comparison result between the format register and the output block format;

mask the first comparison result by an output block format mask; and

generate a second comparison result between the format register and the masked first comparison result, the second comparison result corresponding to the updated format register.

55. (previously presented):

The computer system of claim 52 wherein the program code further causing the processor to:

execute a self-correcting code if the masked input block format is different than the input block format.

56. (previously presented):

The computer system of claim 55 the program code causing the processor to execute causes the processor to:

assert a correction condition based on the format register and the input block format.

57. (previously presented):

The computer system of claim 56 the program code causing the processor to assert causes the processor to:

compare the format register to the input block format if the input block format asserts an access status of the source register.

58. (previously presented):

The computer system of claim 56 wherein the program code further causing the processor to:

convert the source register from format contained in the format register to format contained in the input block format; and

set the format register to the input block format.

59. (previously presented):

The computer system of claim 51 wherein the program code causing the processor to determine the register format causes the processor to:

determine one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.

60. (previously presented):

The computer system of claim 58 wherein the program code causing the processor to detect the block format inconsistency causes the processor to:

detect the block format inconsistency during the execution phase that follows a translation phase in the binary translation.